moments—the only reference to a natural sound world. It is a music that goes beyond our cultural barriers to perception, therefore it demands many listenings.

This double CD is a must if one wants to understand and become familiar with the main body of early electronic pieces. It is important to know the roots and the first attempts to make music on tape, especially in the case of Koenig's music because he leaves many doors open for further research.

III. Software

INTERACTOR 4.0.8

Mark Coniglio and Morton Subotnick, designers. Interactive Music Systems, P. O. Box 6727, Santa Fe, New Mexico 87502-6727, U.S.A. Tel: 213-653-7747.

Reviewed by Robert Rowe, Dept. of Music, New York University, 35 W. Fourth St., Room 777, New York, NY 10003, U.S.A.

Interactor is a graphic MIDI programming environment for Apple Macintosh computers. The environment was designed by Mark Coniglio and Morton Subotnick, and implemented by Coniglio. Interactor programs process several types of events, including MIDI, timing and Macintosh events. Statements in Interactor generally follow an 'if-then' logic flow, in which attributes of incoming events are evaluated to determine whether they match some conditions listed in the 'if' part of a statement. When a condition is found to be true, additional operators (the 'then' part) are executed. Such if-then sequences can occur several times in a single statement; whenever a conditional operator returns a false value, execution of the current statement is ended and the next one initiated. Collections of statements make up 'scenes' and scenes are organized into 'scene groups'. The highest level of organization in the language is the 'document', which comprises up to eight active scene groups.

Figure 1 shows a scene edit window, with one statement made of two *operators*, the basic building blocks of Interactor. Operators are used to evaluate conditions or execute actions. In the example statements, the first operator is a conditional operator looking for any MIDI 'Note On' message. The box below the operator is a comment showing the values of the operator's parameters. These parameters can be modified simply by double-clicking on the operator, which brings up a dialogue box wherein parameter values can be textually modified. In this case, whenever a 'Note On event' arrives, execution will pass on to the next operator in line, from left to right. Then a 'Send Note' operator will be invoked, performing a middle C (C3) on modem channel 1 (M1). Action operators are always true—that is, any additional operators after an action will be executed until a conditional operator tests false or the end of the statement is found.

Figure 2 shows part of the 'Operator Select' window. The 'conditions' class of operators is highlighted and some of the conditional operators are shown, such as 'Test Note On' and 'Test Note Off'. These comments and the dialogue boxes for each operator are the primary on-line means of finding out what each operator does. Some pop-up 'help' windows for the operators would be a useful addition. Note that in Fig. 2 there are two classes of operators labelled 'Seq Play' and 'Seq Rec(ord)'. These operators access Interactor's eight simultaneous multichannel sequences.

Documents can maintain up to 256 tracks of sequenced data. Any of the eight sequences can be assigned to any of the 256 tracks with the ease and flexibility characteristic of the language. Sequence volume, tempo and transposition are modified with custom operators. It is not possible, however, to route the output of the sequence through subsequent Interactor statements. One can route Interactor's sequencer output back to its input through Apple's MIDI Manager, but this workaround is clearly incompatible with the elegance of the overall design.

The language supports eight independent timebases, so that each sequencer can have its own temporal behavior. The timebases are also used by scheduling operators such as the 'Delay Timer,' which will postpone the execution of subsequent operators in a statement by some number of ticks in a timebase. Time is expressed in measures, beats and ticks, where 480 ticks equal one quarter note. This allows tempo to be varied while individual events maintain their relationship to an underlying metric structure. Timebases can be easily and flexibly changed with the appropriate operator, but the Interactor

Fig. 1. An example of a scene edit window. This scene consists of one statement made up of two operators. The bottom two boxes display the parameters for the operators.

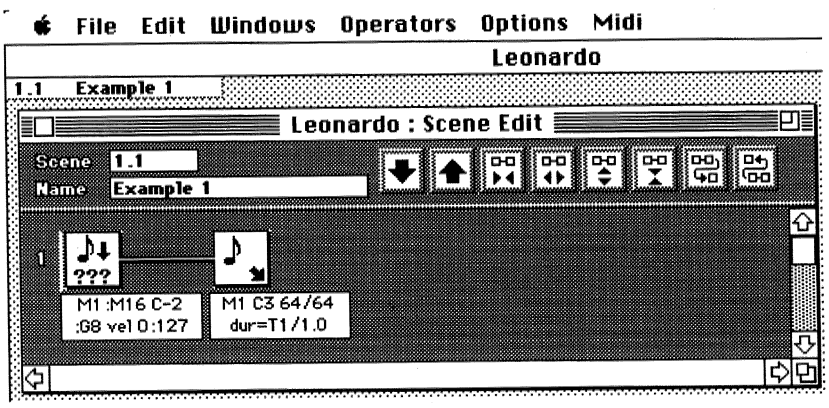
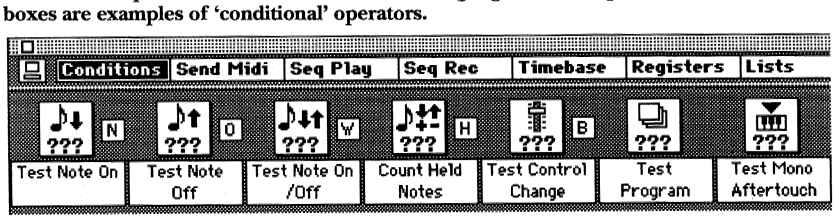


Fig. 2. An example of the 'Operator Select' window. The various classes of operators are listed in the top row, with the 'Conditions' class highlighted. The operators shown in the boxes are examples of 'conditional' operators.



timebase scheme is too limited. There is no provision for a straight millisecond clock, which means that the Delay Timer cannot be set to always delay execution by exactly one second, for example, unless one took care to never alter the tempo of some dedicated timebase. Other timing formats should be supported, such as millisecond and SMPTE counters, as well as synchronization with external clocks through the MIDI manager.

The scene shown in Fig. 1 implements a constant interaction. For any arriving note on message, middle C will be played out. Variable behavior is achieved through use of the environment's 112 internal registers. Registers 1 through 4 are special, in that they will be set to values from incoming events. These values can then be transferred to other registers, used in arithmetic expressions or referenced in the parameter lists for any action operator. A 'Program Change' message could be used to tie a track to a sequencer, for example, by taking the program number from the input register and using that to select a track for a sequencer. The other storage option in Interactor is the 'List'. Values can be stored in lists in a variety of formats and appended, deleted, counted, searched or manipulated in several other ways. Saving a document creates a file containing all its Scenes, Controls, Tracks, Lists and Symbols. If the current values of all registers are saved to a List, the complete state of the program can be saved to a file, allowing 'snapshots' of performance contexts to be maintained as desired.

Interactor is a powerful and flexible environment for the development of interactive music systems. It will be useful for anyone needing to shape custom responses to musical situations. Comparisons will inevitably be drawn with Max, the Opcode graphic MIDI environment that has found widespread use owing to the same motivations that led to Interactor. Certainly there are similarities, particularly in terms of the kinds of events they handle and the power they offer for realizing compositional processes. There are important differences as well: (1) Interactor's left-to-right statement construction enforces a graphic layout that is consistent and clear, (2) the use of registers and lists to convey input information and pass data between operators is a significant departure from Max's patchcords and (3) the combination of sequencers and timebases in Interactor provides powerful support for the interactive performance of sequenced material.

On the other hand, Interactor's most immediate need is to open out onto the outside world. Implementing a greater variety of timing formats and allowing synchronization with external clocks is an important first step. Digital signal-processing operators would be very useful, for processing audio streaming through a Motorola 56000 card, for example, or for playing back sound files. Operators for multimedia performance using Apple's Quicktime would similarly be an obviously useful extension. Some documented scheme allowing users to write their own operators might be the quickest way to expand the environment's functionality. Interactor provides a well-designed, impressively implemented paradigm for developing computer-music performance systems. If the environment can keep pace with the rapidly evolving context of interactive multimedia, it will be a greatly valued tool for years to come.

IV. Bibliography

Recommended Books

Het Apollohuis: Exhibitions, 1980–1990
Het Apollohuis, Tongelresestraat 81, Eindhoven 5613 DB, the Netherlands, 1991.

Art and the Future. A History/Prophecy of the Collaboration between Science, Technology and Art
D. Davis. Praeger, New York, NY, U.S.A., 1965.

Auditory Scene Analysis
Albert S. Bregman. A Bradford Book, MIT Press, Cambridge, MA, U.S.A., 1990. 773 pp., illus. Trade. ISBN: 0-262-02297-4.

Broken Music
Ursula Block and Michael Glasmeir. Berliner Künstlerprogramm des DAAD, Daadgalerie Berlin, Gemeentemuseum Den Haag, Magasin Grenoble, 1989. 278 pp., illus. Paper with record, $40. ISBN: 3-89357-013-6.

Composition et environnements informatiques. Les cahiers de l'ircam, recherche et musique
IRCAM, Paris, 1992. 188 pp., illus. Paper. ISBN: 2-0909487-02-4.

Computer Applications in Music: A Bibliography
Computer Music and Digital Audio Series, Vols. 4 and 10. A-R Editions, Madison, WI, U.S.A., 1988 and 1991.

Computing in Musicology
Walter B. Hewlett and Eleanor Selfridge-Field, eds. A directory of research published annually by the Center for Computer-Assisted Research in the Humanities, 525 Middlefield Road, Suite 120, Menlo Park, CA, 94025, U.S.A.
E-mail:<XB.L36@Stanford.Bitnet>

Current Directions in Computer Music Research
M. V. Mathews and J. R. Pierce, eds. MIT Press, Cambridge, MA, U.S.A., 1989.

A Dictionary of Electronic and Computer Music Technology
Richard Dobson. Oxford Univ. Press, Oxford, 1992. 224 pp., illus. Trade, $39.95. ISBN: 0-19-311344-9.

Dokumentation Elektroakustischer Musik in Europa
Golo Foellmer, Roland Franck and Folmar Hein. Produced by Technische Universität Berlin, Elektronisches Studio, Deutsche Sektion der internationalen Gesellschaft für Elektroakustische Musik (DecimE); published by Inventionen '92, 1992. 370 pp. Paper. Available from TU Berlin, H51, Dokumentation, Str. des 17. Juni 135, D-1000 Berlin 12, Germany.
E-mail: <hein@mvax.kgw.tu-berlin.de>
Fax: (49) 30-314 21143.

Echo: The Images of Sound
Het Apollohuis, Tongelresestraat 81, Eindhoven 5613 DB, the Netherlands, 1986.

Echo II
Het Apollohuis, Tongelresestraat 81, Eindhoven 5613 DB, the Netherlands, 1991. Documentation and CD.

Elements of Computer Music
F. R. Moore. Prentice Hall, Englewood Cliffs, NJ, U.S.A., 1990.

Extended Musical Interface with the Human Nervous System: Assessment and Prospectus
David Rosenboom. Leonardo Monographs, International Society for Art, Science and Technology, San Francisco, CA, U.S.A., 1989.

Film Music—A Neglected Art: A Critical Study of Music in Films
Roy M. Prendergast. Second edition. W. W. Norton, New York, NY, U.S.A., 1992. 329 pp., illus. Paper, $10.95. ISBN: 0-393-30874-X.

Foundations of Computer Music
C. Roads and J. Strawn, eds. MIT Press, Cambridge, MA, U.S.A., 1985.

Live Electronics
Peter Nelson and Stephen Montague, eds. Contemporary Music Review, Vol. 6, No. 1, Harwood Academic Publishers, Reading, U.K., 1991. 237 pp., illus. Paper. ISBN: 3-7186-5116-5; ISSN: 0749-4467.

Materials and Techniques of Twentieth-Century Music
Stefan Kostka. Prentice-Hall, Englewood Cliffs, NJ, U.S.A., 1989. 337 pp., illus. Trade. ISBN: 0-13-560830-9.